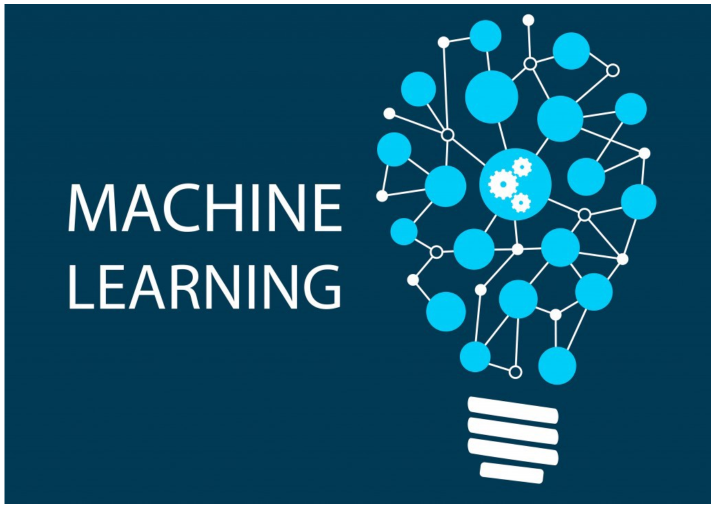
An update on scikit-learn

0.20 and beyond

Andreas Müller
Columbia University, scikit-learn

What is scikit-learn?

python ♥

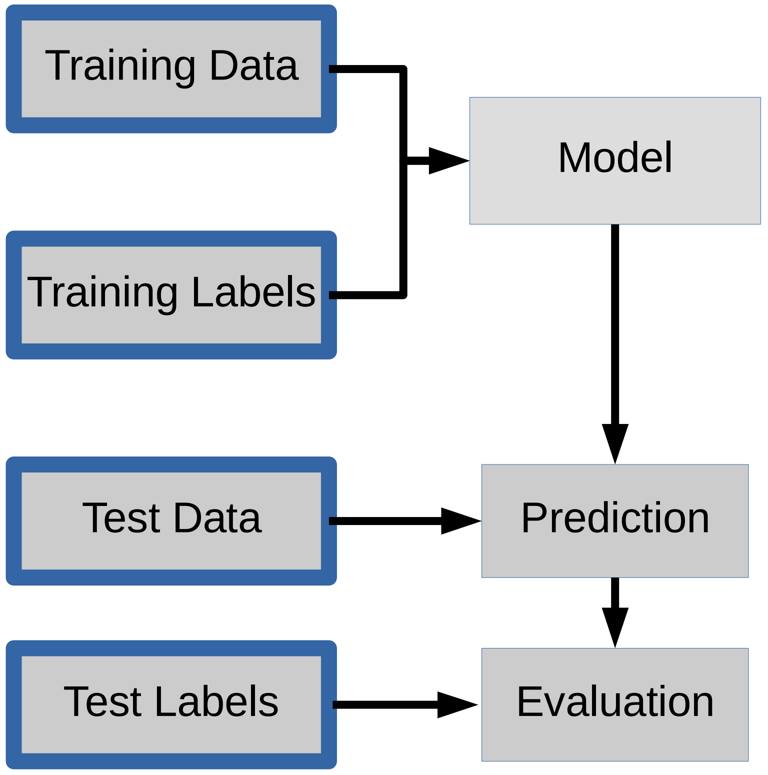


Basic API review

Representing Data

$$X = \begin{array}{rl}
\text{one sample} & \left(\begin{array}{ccccc}
1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\
6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\
2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\
1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\
0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\
5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\
3.7 & 7.8 & 2.6 & 3.2 & 6.3
\end{array}\right) \\
 & \text{one feature}
\end{array}
\qquad
y = \begin{array}{c}
\left(\begin{array}{c}
1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7
\end{array}\right) \\
\text{outputs / labels}
\end{array}$$

```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

clf.score(X_test, y_test)
```



Basic API

`estimator.fit(X, [y])`

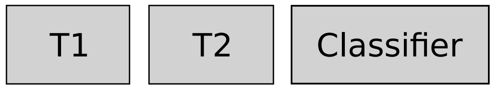
`estimator.predict`	**`estimator.transform`**
Classification	Preprocessing
Regression	Dimensionality reduction
Clustering	Feature selection
	Feature extraction

Cross -Validated Grid Search

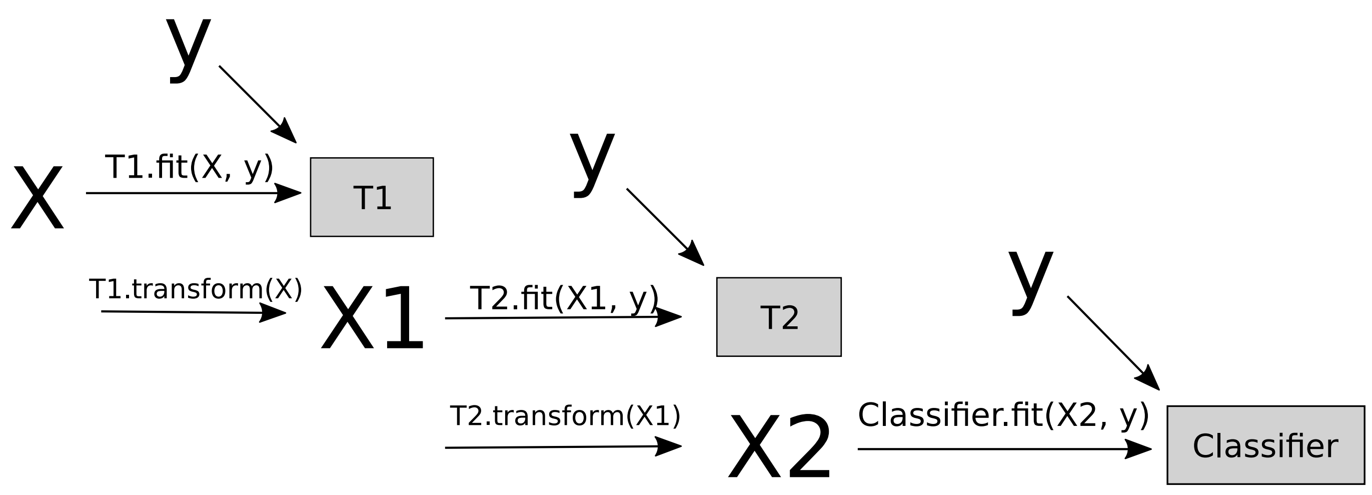
```
from sklearn.grid_search import GridSearchCV
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

param_grid = {'C': 10. ** np.arange(-3, 3),
              'gamma': 10. ** np.arange(-3, 3)}
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
grid.predict(X_test)
grid.score(X_test, y_test)
```

Pipelines

```
pipe = make_pipeline(T1(), T2(), Classifier())
```



```
pipe.fit(X, y)
```



```
pipe.predict(X')
```

$$\begin{tikzcd} X' \arrow[r, "\text{T1.transform(X')}"] & X'1 \arrow[r, "\text{T2.transform(X'1)}"] & X'2 \arrow[r, "\text{Classifier.predict(X'2)}"] & y' \end{tikzcd}$$

Pipelines

```python
from sklearn.pipeline import make_pipeline

pipe = make_pipeline(StandardScaler(), SVC())
pipe.fit(X_train, y_train)
pipe.predict(X_test)
```

```python
param_grid = {'svc__C': 10. ** np.arange(-3, 3),
              'svc__gamma': 10. ** np.arange(-3, 3)}

scaler_pipe = make_pipeline(StandardScaler(), SVC())
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
```

scikit learn 0.20

```
conda install scikit-learn=0.20rc1 -c conda-forge/label/rc -c conda-forge
```

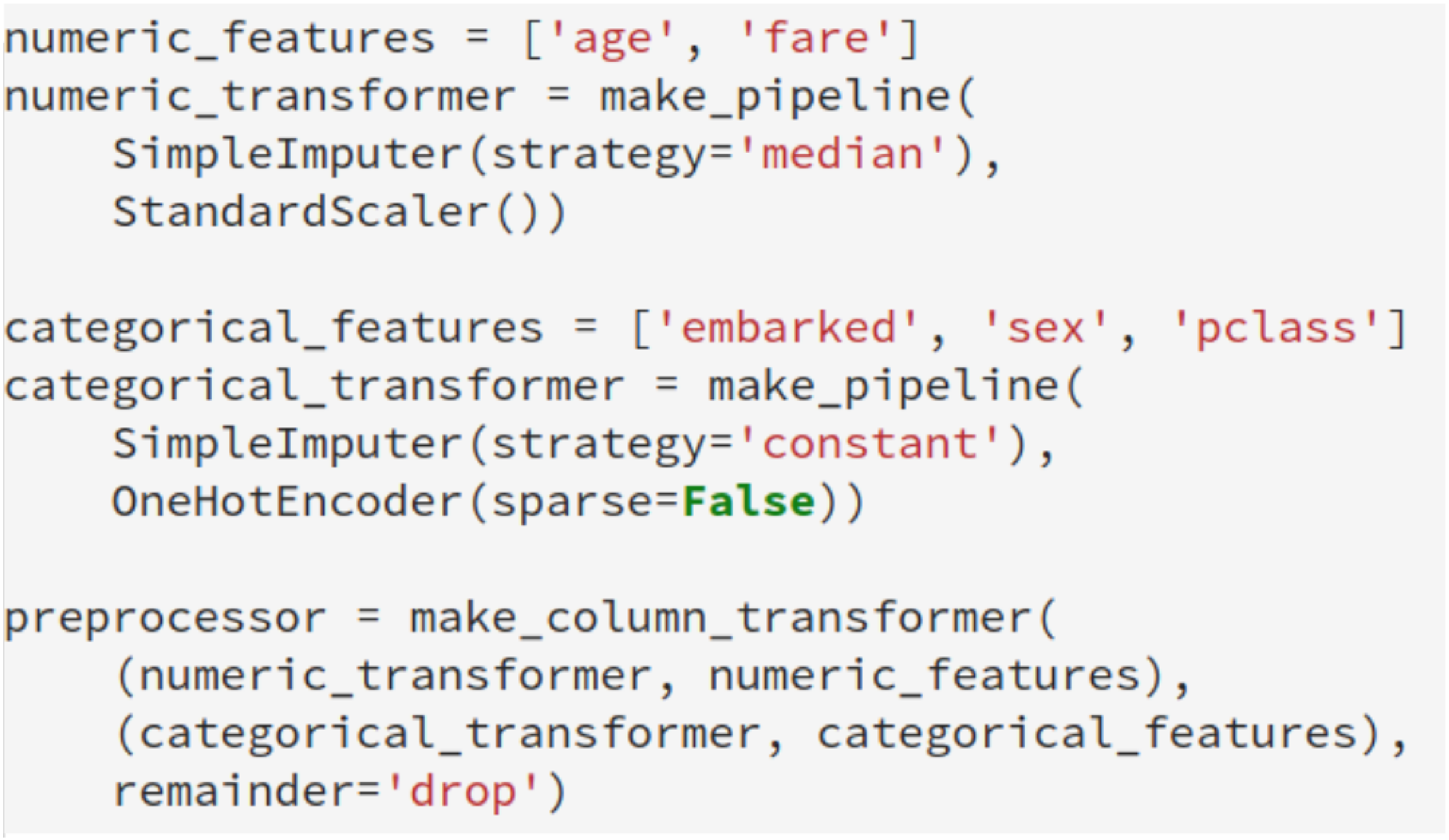
```
pip install --pre scikit-learn
```

OneHotEncoder for Strings

```
>>> from sklearn.preprocessing import OneHotEncoder
>>> enc = OneHotEncoder(handle_unknown='ignore')
>>> X = [['Male', 1], ['Female', 3], ['Female', 2]]
>>> enc.fit(X)
```

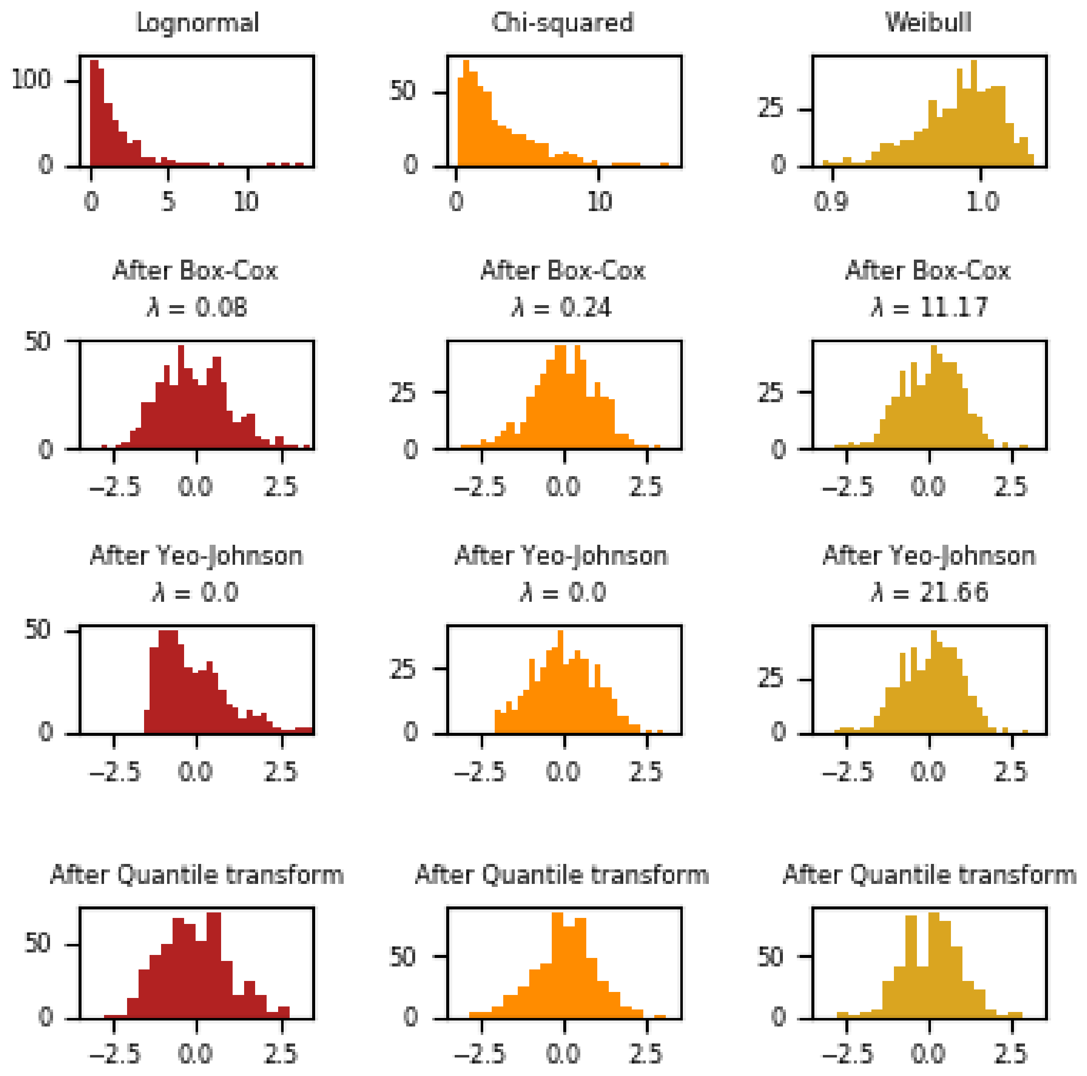
```
>>> enc.transform([['Female', 1], ['Male', 4]]).toarray()
array([[1., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0.]])
>>> enc.inverse_transform([[0, 1, 1, 0, 0], [0, 0, 0, 1, 0]])
array([['Male', 1],
       [None, 2]], dtype=object)
```

ColumnTransformer



```python
numeric_features = ['age', 'fare']
numeric_transformer = make_pipeline(
    SimpleImputer(strategy='median'),
    StandardScaler())

categorical_features = ['embarked', 'sex', 'pclass']
categorical_transformer = make_pipeline(
    SimpleImputer(strategy='constant'),
    OneHotEncoder(sparse=False))

preprocessor = make_column_transformer(
    (numeric_transformer, numeric_features),
    (categorical_transformer, categorical_features),
    remainder='drop')
```

PowerTransformer



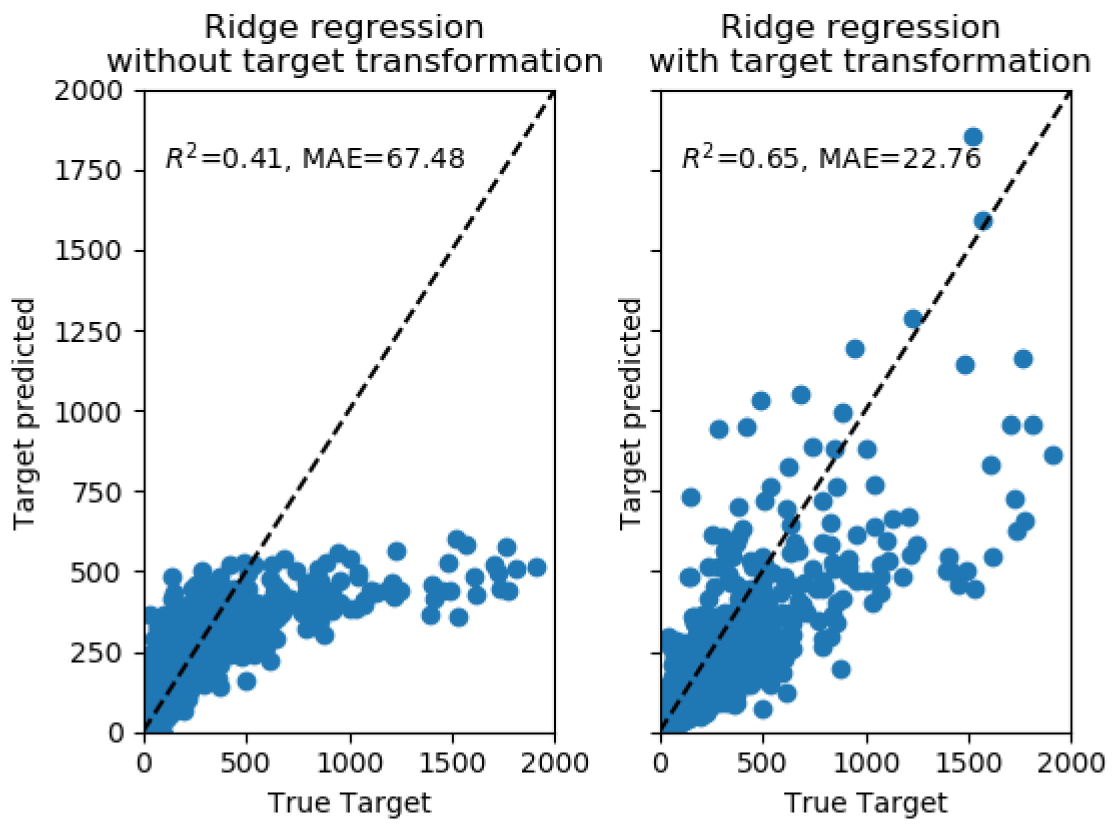
$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(y_i) & \text{if } \lambda = 0, \end{cases}$$

Missing Value treatment

- Scalers
- SimpleImputer
- MissingIndicator

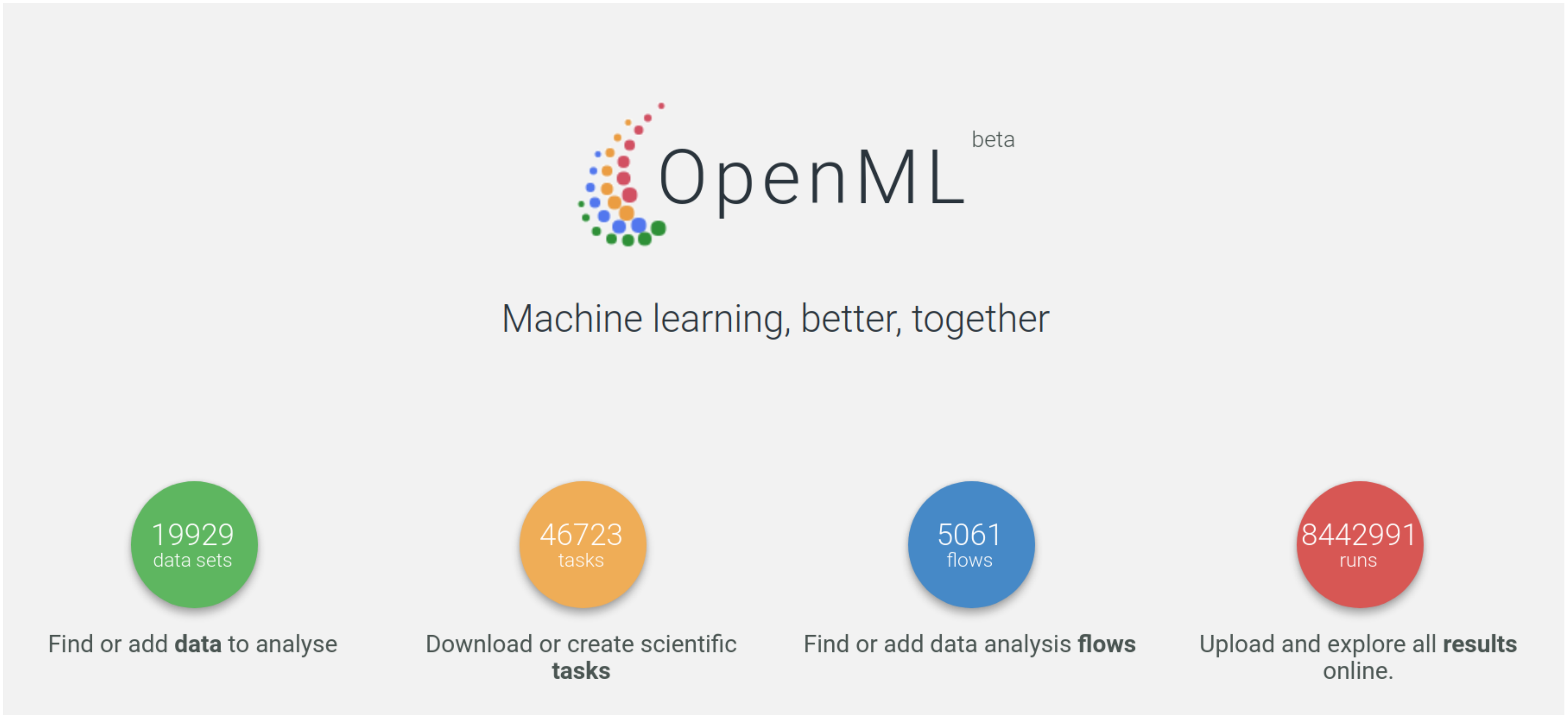
TransformedTargetRegressor

```
regr_trans = TransformedTargetRegressor(regressor=RidgeCV(),
                                        func=np.log1p,
                                        inverse_func=np.expm1)
regr_trans.fit(X_train, y_train)
y_pred = regr_trans.predict(X_test)
```



Synthetic data

OpenML Dataset Loader


OpenML beta
Machine learning, better, together
19929 data sets
46723 tasks
5061 flows
8442991 runs
Find or add **data** to analyse
Download or create scientific **tasks**
Find or add data analysis **flows**
Upload and explore all **results** online.

```
>>> from sklearn.datasets import fetch_openml
>>> mice = fetch_openml('miceprotein', version=4, data_home=custom_data_home)
```

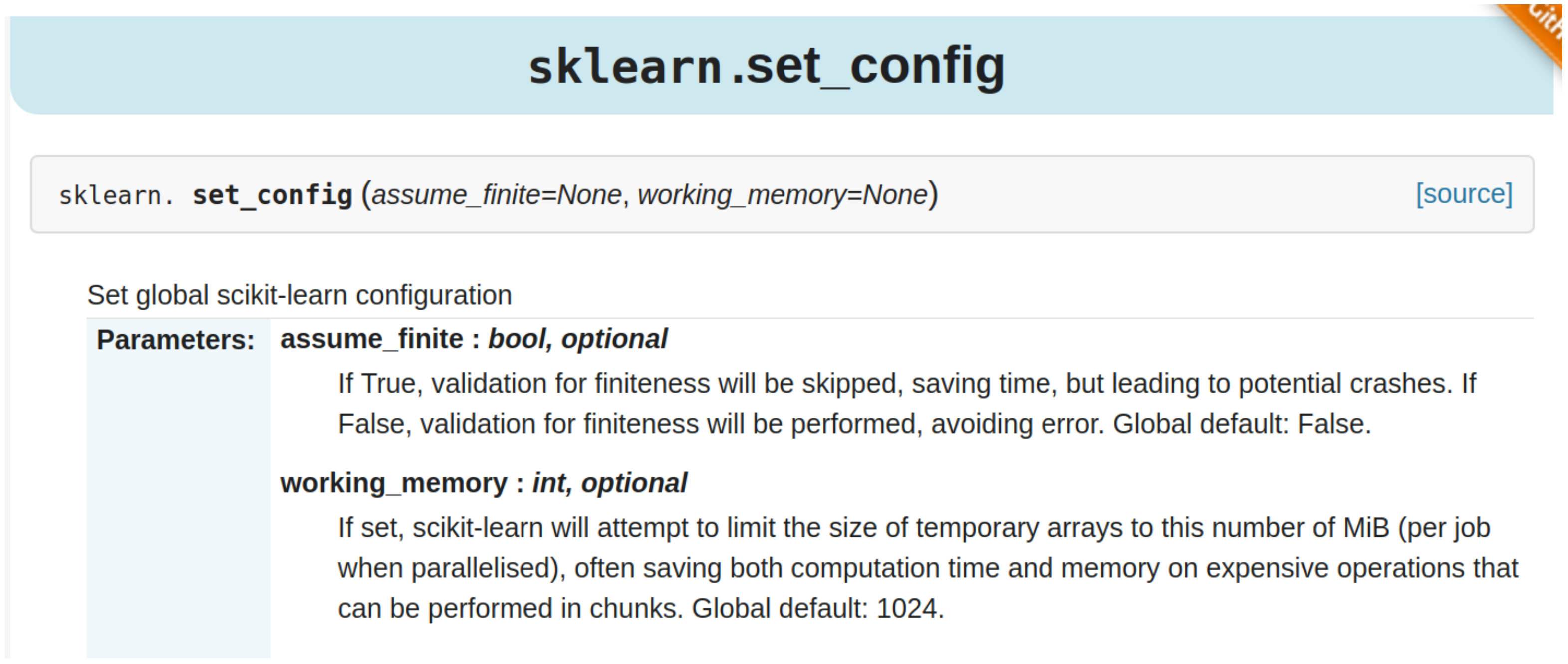
Loky
A Robust and reusable Executor

https://loky.readthedocs.io/en/stable/

- An alternative for multiprocessing.pool.Pool and concurrent.futures.ProcessPoolExecutor
- No need for if __name__ == "__main__": in scripts
- Deadlock free implementation
- Consistent spawn behavior
- No random crashes with odd BLAS / OpenMP libraries

Global config and working memory

```
>>> import sklearn
>>> with sklearn.config_context(working_memory=128):
...     pass  # do chunked work here
```



sklearn.set_config

`sklearn.set_config(assume_finite=None, working_memory=None)` [source]

Set global scikit-learn configuration

Parameters:

assume_finite : *bool, optional*

If True, validation for finiteness will be skipped, saving time, but leading to potential crashes. If False, validation for finiteness will be performed, avoiding error. Global default: False.

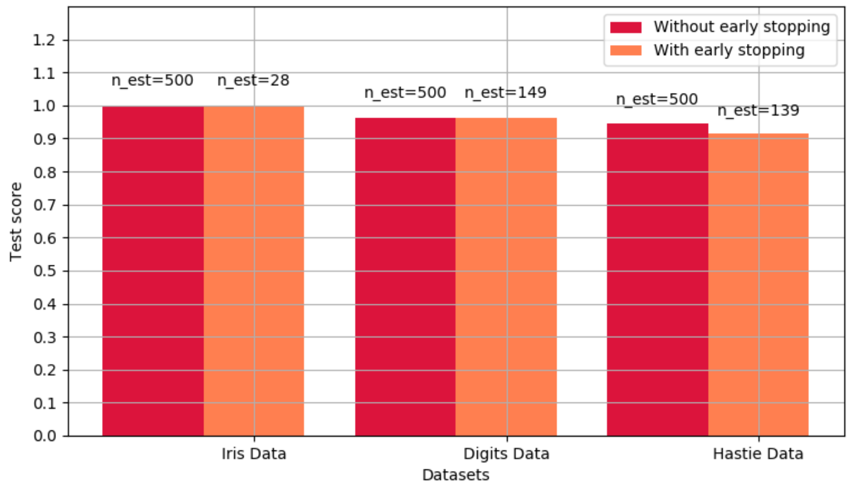
working_memory : *int, optional*

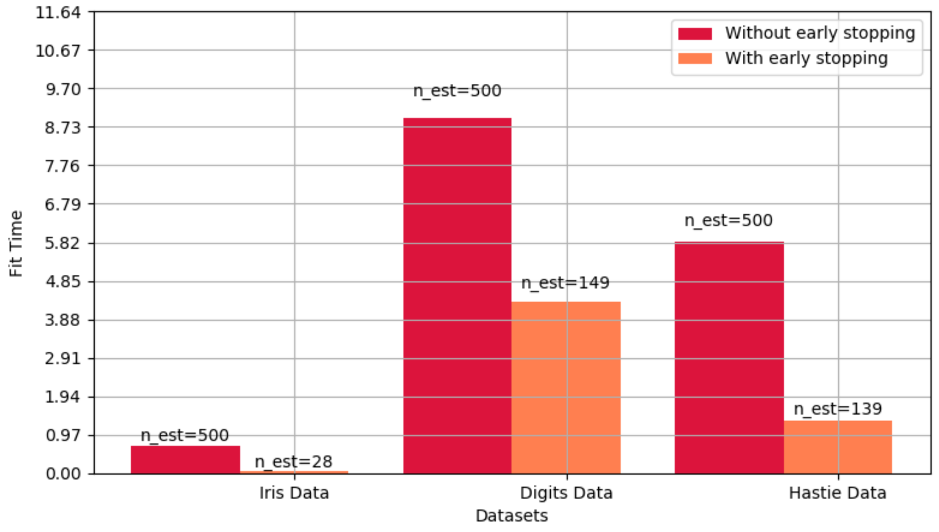
If set, scikit-learn will attempt to limit the size of temporary arrays to this number of MiB (per job when parallelised), often saving both computation time and memory on expensive operations that can be performed in chunks. Global default: 1024.

Gradient Boosting Early Stopping

```
GradientBoostingClassifier(n_estimators=n_estimators,
                           validation_fraction=0.2,
                           n_iter_no_change=5, tol=0.01,
                           random_state=0)
```

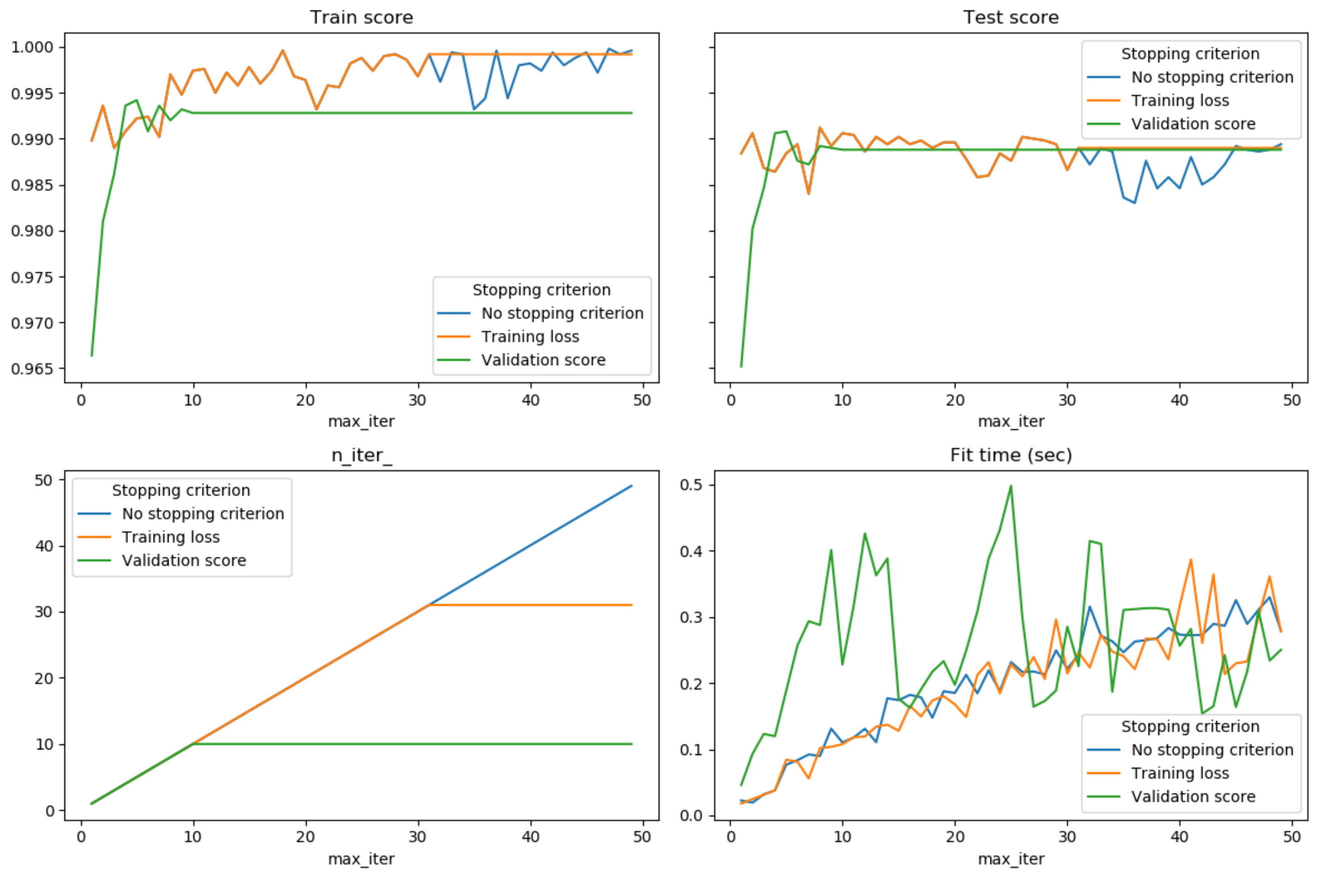
Activated by “n_iter_no_change”

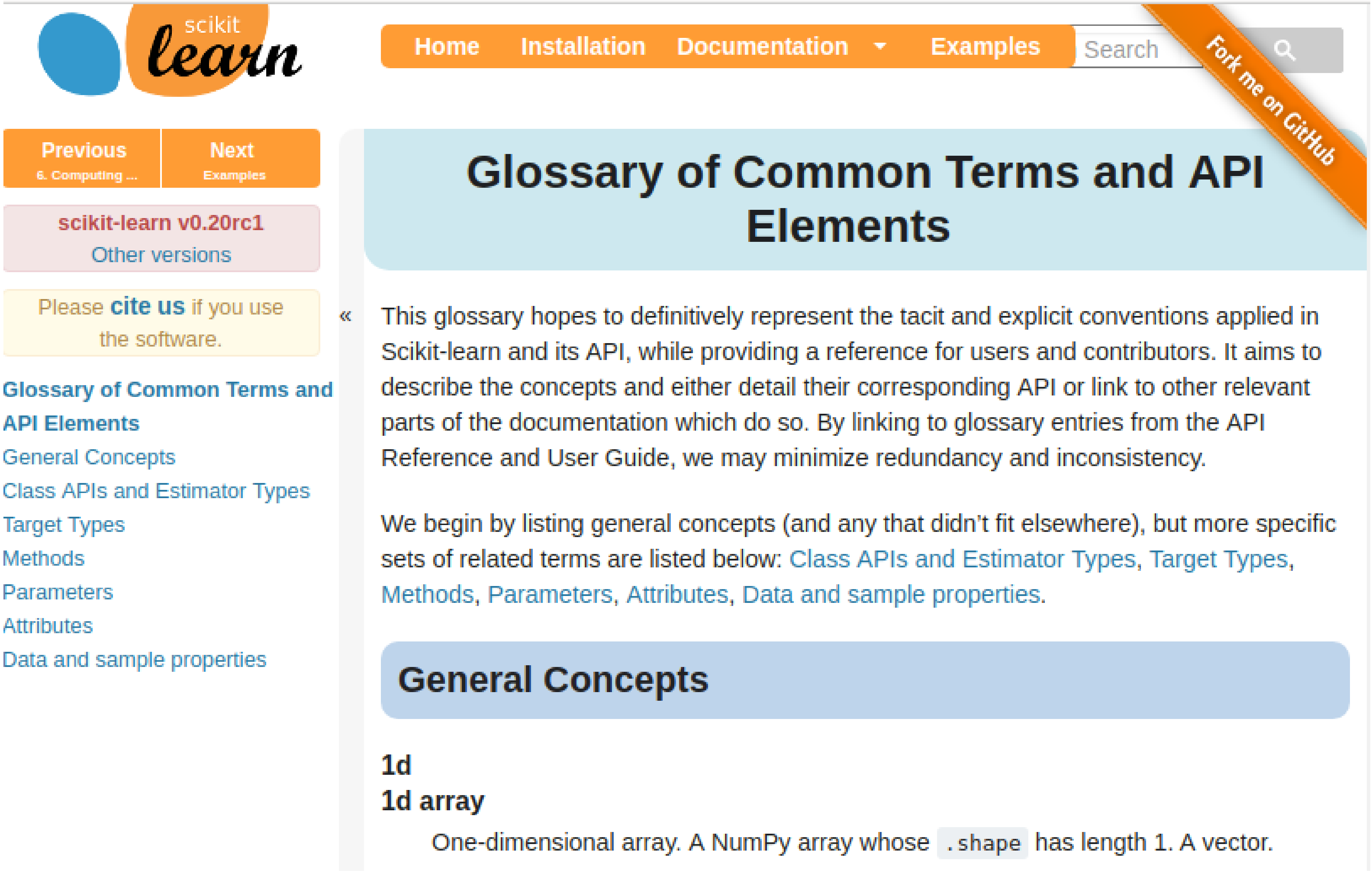




SGD et al early stopping

```
'Training loss':
linear_model.SGDClassifier(early_stopping=False, n_iter_no_change=3,
                           tol=0.1),
'Validation score':
linear_model.SGDClassifier(early_stopping=True, n_iter_no_change=3,
                           tol=0.0001, validation_fraction=0.2)
```





Glossary of Common Terms and API Elements

This glossary hopes to definitively represent the tacit and explicit conventions applied in Scikit-learn and its API, while providing a reference for users and contributors. It aims to describe the concepts and either detail their corresponding API or link to other relevant parts of the documentation which do so. By linking to glossary entries from the API Reference and User Guide, we may minimize redundancy and inconsistency.

We begin by listing general concepts (and any that didn't fit elsewhere), but more specific sets of related terms are listed below: Class APIs and Estimator Types, Target Types, Methods, Parameters, Attributes, Data and sample properties.

General Concepts

1d
1d array

One-dimensional array. A NumPy array whose `.shape` has length 1. A vector.

Better defaults

All random forests:
n_estimators from 10 to 100 (in 0.22)

Cross-validation:

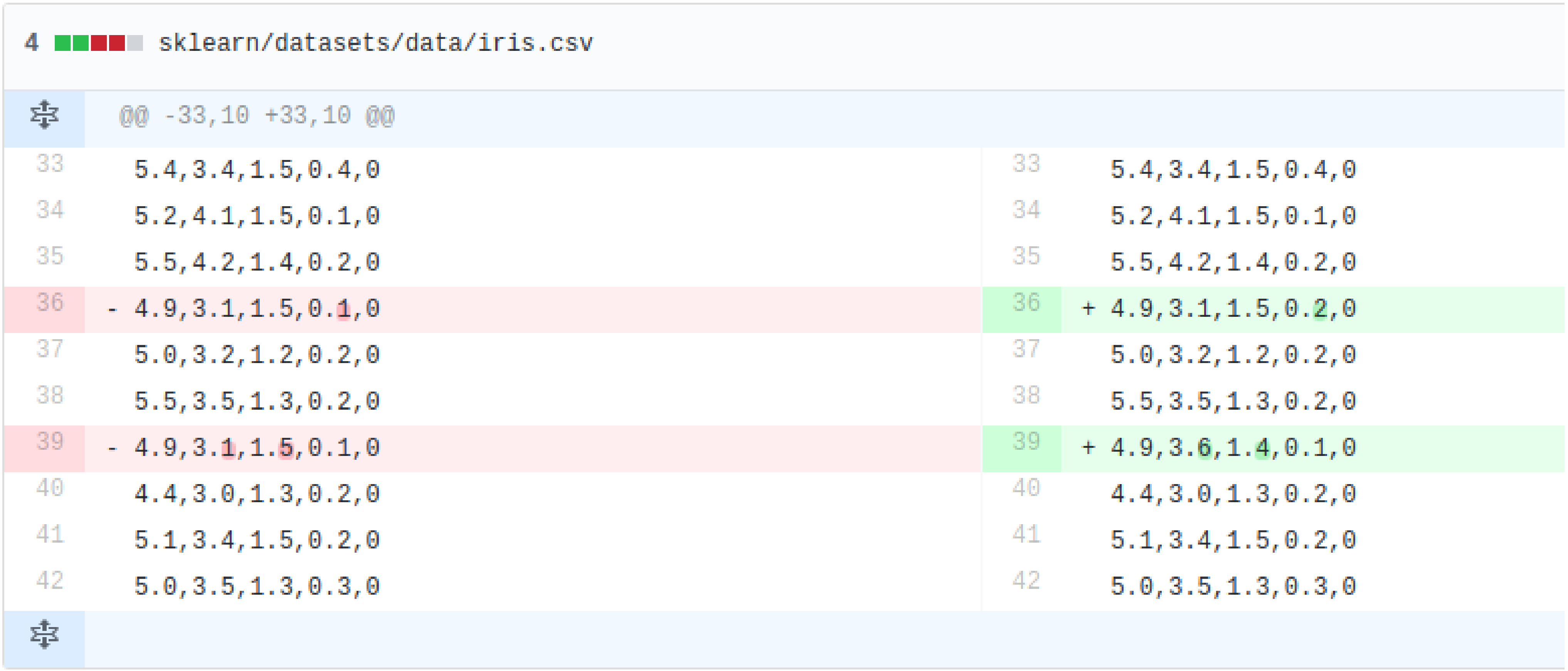
cv from 3 to 5 (in 0.22)

Grid-Search:
iid to False (in 0.22)
Remove iid (in 0.24)

LogisticRegression defaults

- solver=’lbfgs’ (from ‘liblinear’)
- multiclass=’auto’ (from ‘ovr’)

“Fixed” iris



```
4 🟩🟩🟥🟥⬜ sklearn/datasets/data/iris.csv

@@ -33,10 +33,10 @@
  5.4,3.4,1.5,0.4,0                    5.4,3.4,1.5,0.4,0
  5.2,4.1,1.5,0.1,0                    5.2,4.1,1.5,0.1,0
  5.5,4.2,1.4,0.2,0                    5.5,4.2,1.4,0.2,0
- 4.9,3.1,1.5,0.1,0                  + 4.9,3.1,1.5,0.2,0
  5.0,3.2,1.2,0.2,0                    5.0,3.2,1.2,0.2,0
  5.5,3.5,1.3,0.2,0                    5.5,3.5,1.3,0.2,0
- 4.9,3.1,1.5,0.1,0                  + 4.9,3.6,1.4,0.1,0
  4.4,3.0,1.3,0.2,0                    4.4,3.0,1.3,0.2,0
  5.1,3.4,1.5,0.2,0                    5.1,3.4,1.5,0.2,0
  5.0,3.5,1.3,0.3,0                    5.0,3.5,1.3,0.3,0
```

A note on deprecations ...

[WIP]

Open

Work In Progress

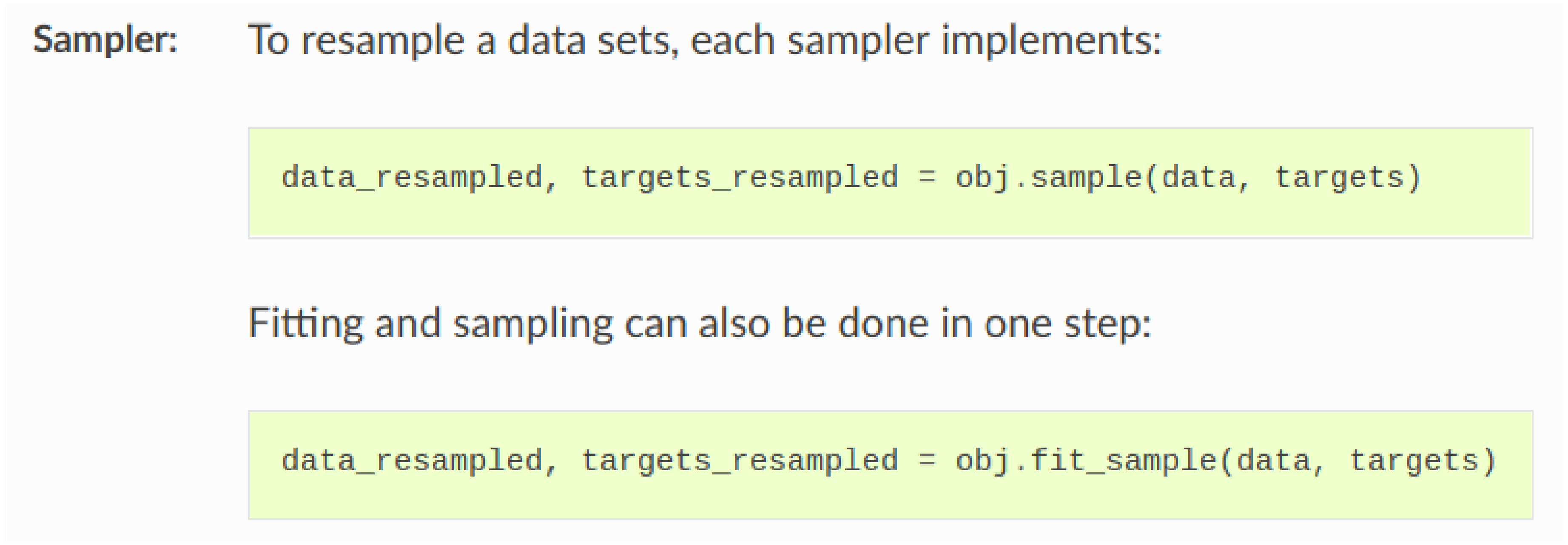
A (draft) Roadmap

https://github.com/scikit-learn/scikit-learn/wiki/Draft-Roadmap-2018

Dropping 2.7 (and 3.4!)

Loky / OpenMP

Imbalanced-learn integration



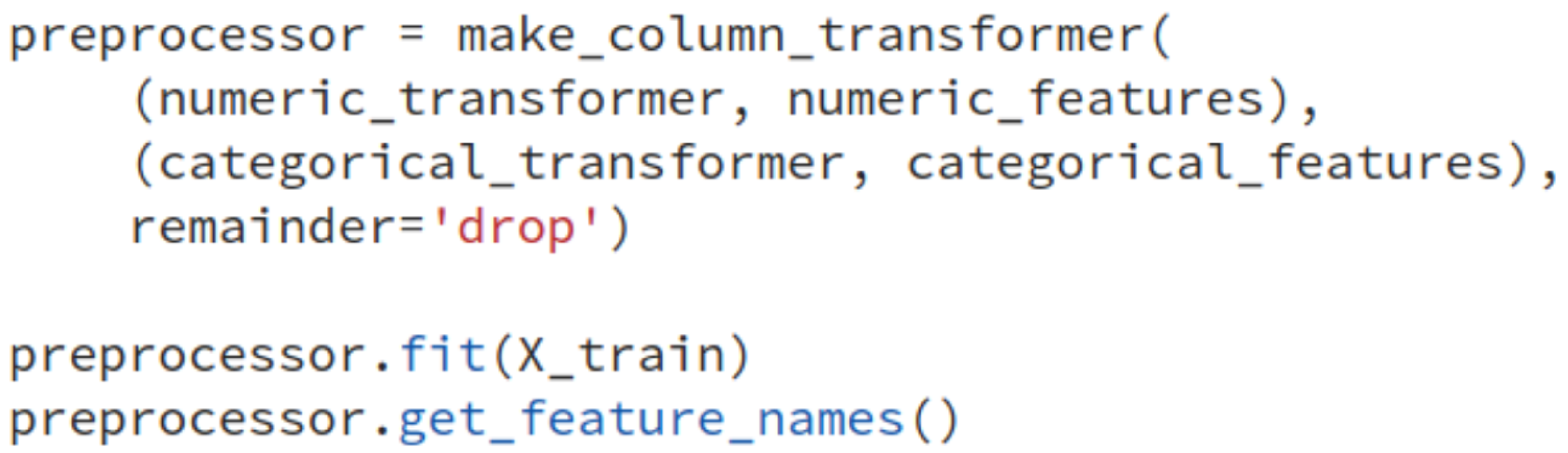
Sampler: To resample a data sets, each sampler implements:

```
data_resampled, targets_resampled = obj.sample(data, targets)
```

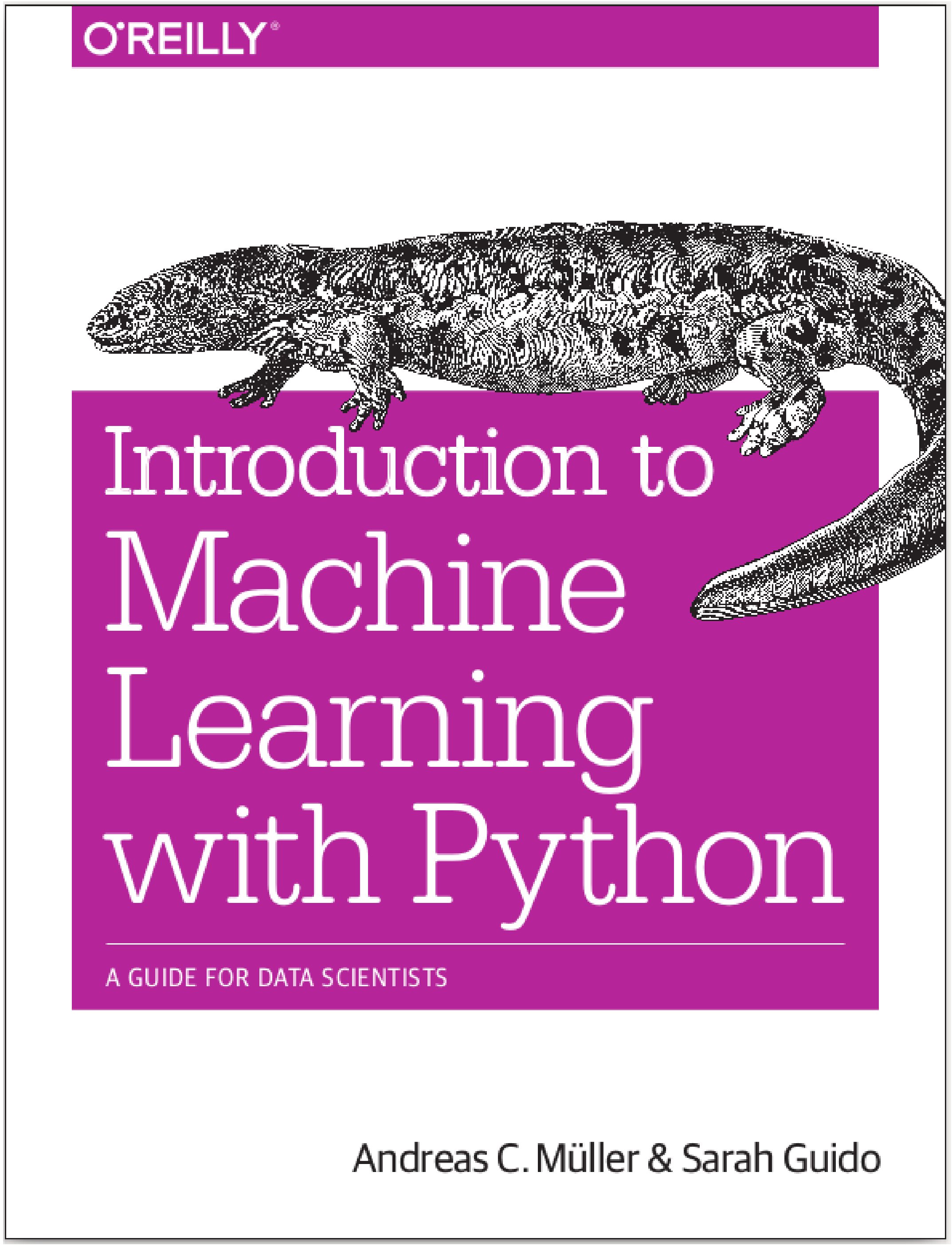
Fitting and sampling can also be done in one step:

```
data_resampled, targets_resampled = obj.fit_sample(data, targets)
```

Pandas & Feature Names



```
preprocessor = make_column_transformer(
    (numeric_transformer, numeric_features),
    (categorical_transformer, categorical_features),
    remainder='drop')

preprocessor.fit(X_train)
preprocessor.get_feature_names()
```


O'REILLY
Introduction to Machine Learning with Python
A GUIDE FOR DATA SCIENTISTS
Andreas C. Müller & Sarah Guido

🌐	amueller.github.io
🐦	@amuellerml
🐱	@amueller
✉	t3kcit @gmail.com